- ***Introducing Debug***

  A debugger is a tool that displays the contents of memory and lets you view registers and variables as they change. You can step through a program one line at a time (called tracing), making it easier to find logic errors. We can Debug to test assembly instructions, try out new programming ideas, or to carefully step through your programs. It takes supreme overconfidence to write an assembly language program and run it directly from DOS (Disk Operating System). For example any call or jump to a location outside your program will almost surely cause the program to crash. For this reason, we would be wise to run any new program we have written in Debug. The Trace the program one line at a time, and watch the stack pointer (SP) very closely as we step through the program, and note any unusual changes to the CS and IP registers.

- ***Debugging functions:***

  - Assemble short programs
  - View a program's source code along with the machine code
  - View the CPU registers and flags
  - Trace or execute a program, watching variables for changes
  - Enter a new value into memory
  - Search for binary or ASCII values in memory
  - Move a block of memory from one location to another
  - Fill a block of memory
  - Load and write disk files and sectors.

All above functions can be achieved by using some debugger commands; we will study them over our Lab lectures.

- ***How to run Debug***

Debug command summary

Debug commands may be divided into four groups: program creation/debugging, memory manipulation, miscellaneous, and input-output.

Program creation and debugging

A	Assemble a program using instruction mnemonics
G	Execute the program currently in memory
R	Display the contents of registers and flags
P	Proceed past an instruction, procedure, or loop
T	Trace a single instruction
U	Disassemble memory into assembler mnemonics

Memory manipulation

C	Compare one memory range with another
D	Dump (display) the contents of memory
E	Enter bytes into memory
F	Fill a memory range with a single value
M	Move bytes from one memory range to another
S	Search a memory range for specific value(s)

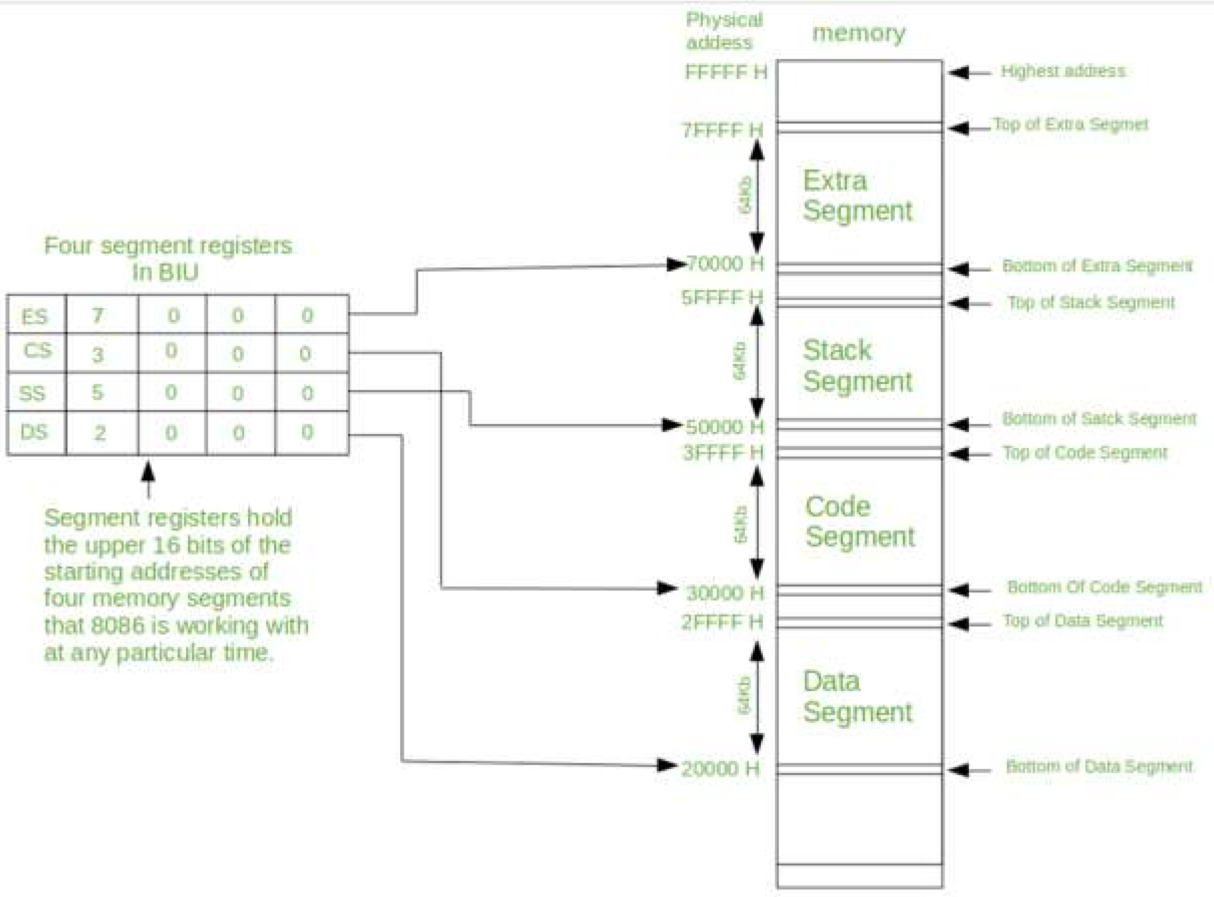
Miscellaneous

H	Perform hexadecimal addition and subtraction
Q	Quit debug and return to DOS

Input-Output

I	Input a byte from a port
L	Load data from disk
O	Send a byte to a port
N	create a filename for use by the L and W commands
W	Write data from memory to disk

- ***Explain the four Segments (CS, DS,ES, and SS)***



- ***D (Dump)***

The D command displays memory on the screen as single bytes in both hexadecimal and ASCII.

Command formats:

D
D address
D range

If no address or range is given, the location begins where the last D command left off, or at location DS:0 if the command is begin typed for the first time. If the address is specified, it consists of either a segment-offset address or just a 16-bit offset. Range consists of the beginning and ending address to dump.

Example	Description
D F000:0	Segment-offset

```
D ES:100      Segment register-offset
D 100         Offset
```

The default segment is DS, so the segment value many be left out unless we want to dump an offset from another segment location. A range may be given, telling Debug to dump all bytes within the range:

```
D 150  15A   (Dump DS:0150 through 015A)
```

Other segment registers or absolute addresses may be used as the following examples